Carnegie Mellon
Software Engineering Institute

Software Product Lines

Paul Clements
Linda Northrop
Product Line Systems Program

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

This work is sponsored by the U.S. Department of Defense.

Carnegie Mellon
Software Engineering Institute

Today's Talk

Introduction

Product Line Concepts
- What
- Why
- How

Conclusion

Carnegie Mellon
Software Engineering Institute

Software Engineering Institute

Applied R&D laboratory situated as a college-level unit at Carnegie Mellon University, Pittsburgh, PA, USA

Established in 1984

Technical staff of 335

Offices in Pittsburgh, Pennsylvania (USA), Arlington, Virginia (USA) and Frankfurt Germany

Purpose: Help others improve their software engineering practices

Carnegie Mellon
Software Engineering Institute

Product Line Systems Program

Our Goal: To enable widespread product line practice through architecture-based development

Carnegie Mellon
Software Engineering Institute

Our Strategy

Software Architecture

(Architecture Tradeoff Analysis Initiative)

Software Product Lines

(Product Line Practice Initiative)

Component Technology

(Predictable Assembly from Certifiable Components Initiative)

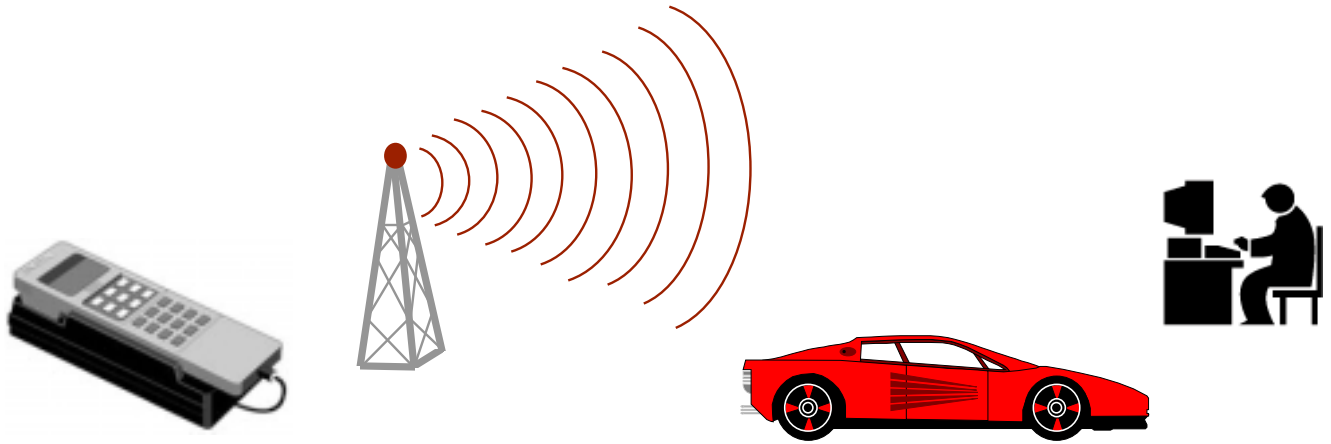
Carnegie Mellon
Software Engineering Institute

Our Customers and Collaborators

ABB
Daimler Chrysler
Caterpillar
Robert Bosch Co.
Raytheon
Foliage
RIM
Unisys
Visteon
LLNL
EPA
FAA
NASA: JSC
NASA: KSC
NASA Goddard
USCG
NRO/CCT
JNIC
DMSO
US Army SOA: TAPO
US Army: FBCB2, CECOM, ATSC, FCS
US Navy: TENA, DDX
US Navy: DDX
US Air Force: F-22, ESC

Philips
Lucent
AT&T
Hewlett Packard
Thomson-CSF
Ericsson
Raytheon
Siemens
Schlumberger
Nokia
Telesoft S.p.A.
Boeing
CelsiusTech
Buzzeo
ALLTEL
Motorola
Cummins, Inc.
General Motors
Lockheed Martin
Salion, Inc.
MarketMaker

Business Success Requires Software Prowess



Software pervades every sector.
Software has become the bottom line for many organizations who never envisioned themselves in the software business.

Carnegie Mellon
Software Engineering Institute

Universal Business Goals

High quality

Quick time to market

Effective use of limited resources

Product alignment

Low cost production

Low cost maintenance

Mass customization

Mind share

improved efficiency and productivity

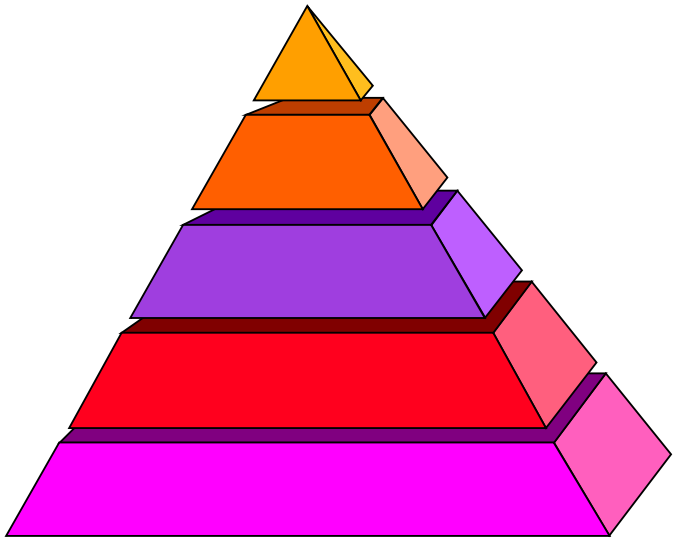
The Ultimate Universal Goal

Software (System) Strategies

Process Improvement

Technology Innovation

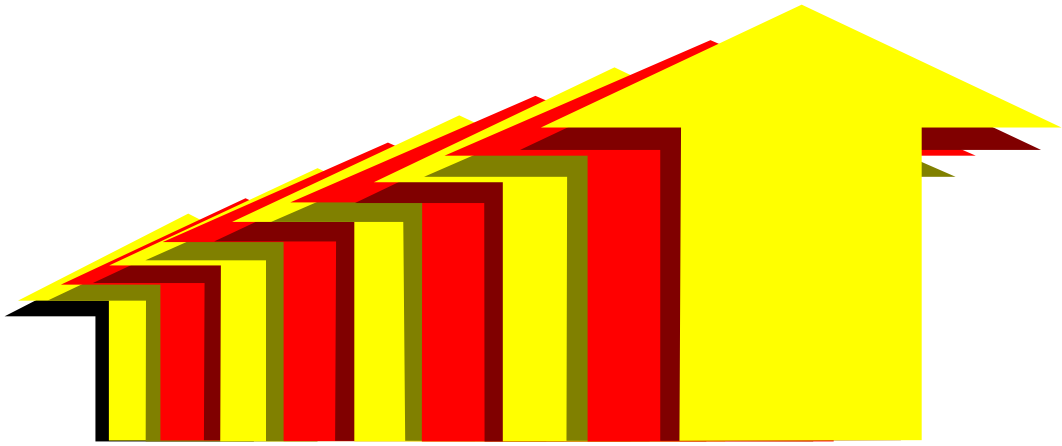
Reuse



Few Systems Are Unique

Most organizations produce families of similar systems, differentiated by features.

Reuse History



1960's Subroutines

1970's Modules

1980's Objects

1990's Components

Focus was small-grained and opportunistic.
Results fell short of expectations.

Imagine Strategic Reuse

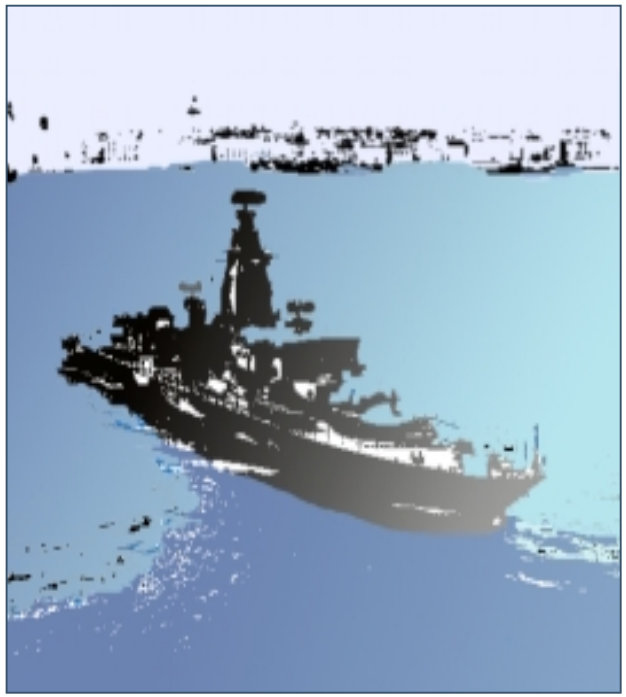
strategic reuse

Carnegie Mellon
Software Engineering Institute

CelsiusTech: Ship System 2000

A family of 55 ship systems

Integration test of 1-1.5 million SLOC requires 1-2 people
Rehosting to a new platform/OS takes 3 months
Cost and schedule targets are predictably met
Performance/distribution behavior are known in advance
Customer satisfaction is high
Hardware-to-software cost ratio changed from 35:65 to 80:20



Carnegie Mellon
Software Engineering Institute

Cummins Inc.: Diesel Engine Control Systems

Over 20 product groups with over 1000 separate engine applications

Product cycle time was slashed from 250 person-months to a few person-months
Build and integration time was reduced from one year to one week
Quality goals are exceeded
Customer satisfaction is high
Product schedules are met

Carnegie Mellon
Software Engineering Institute

National Reconnaissance Office / Raytheon: Control Channel Toolkit

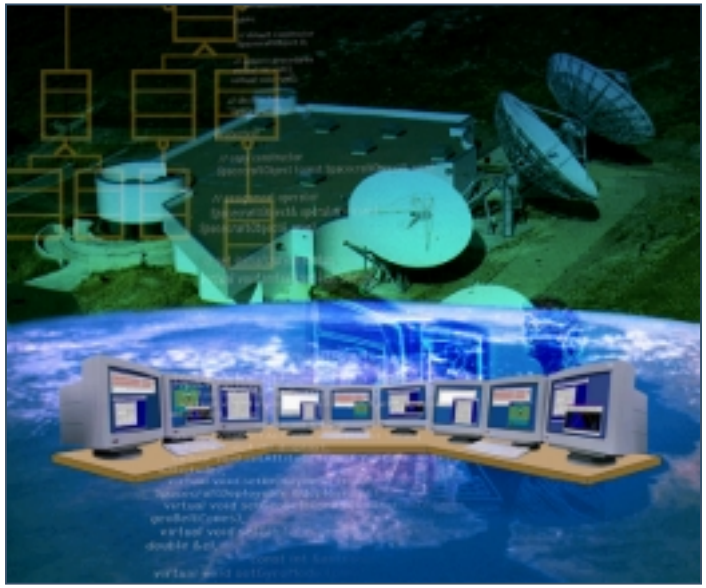
Ground-based spacecraft command and control systems

Increased quality by 10X
Incremental build time
- reduced from months to weeks

Software productivity
- increased by 7X

Development time and costs
- decreased by 50%

Decreased product risk



Market Maker GmbH: MERGER

Internet-based stock market software

Each product “uniquely” configured

Three days to put up a customized system

Nokia Mobile Phones

Product lines with 25-30 new products per year

Across products there are

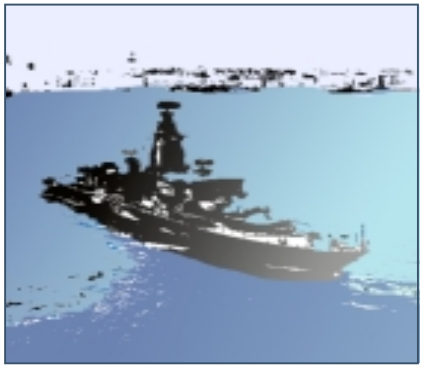
- varying number of keys
- varying display sizes
- varying sets of features
- 58 languages supported
- 130 countries served
- multiple protocols
- needs for backwards compatibility
- configurable features
- needs for product behavior change after release

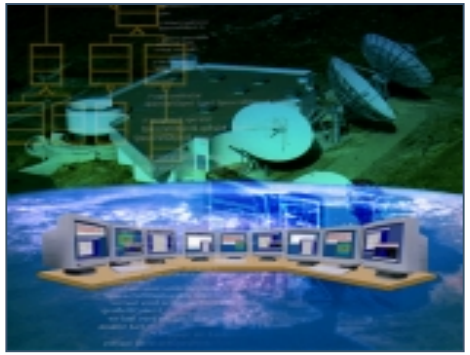
How Did They Do It?

strategic reuse

employed to achieve explicit business goals

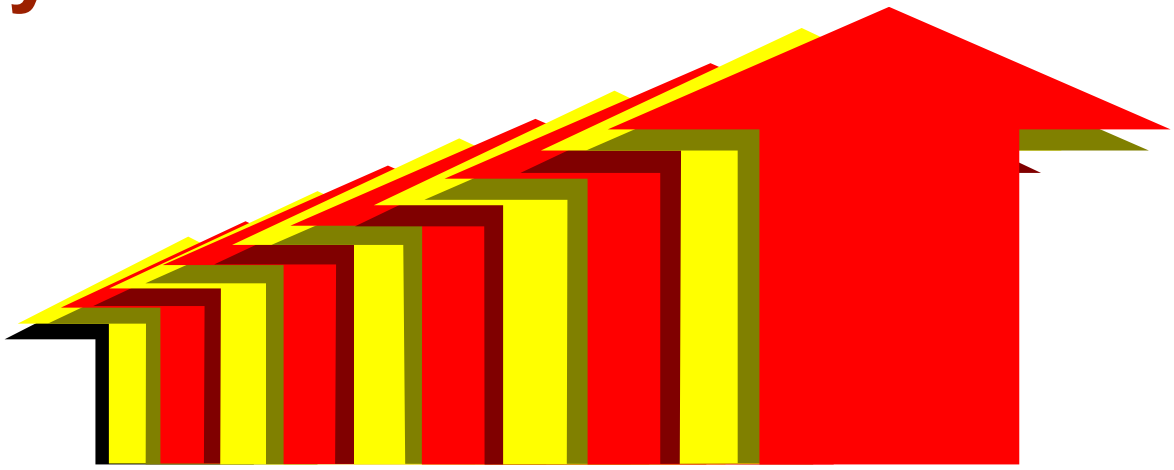
Strategic Reuse is Different





Software Product Lines

Reuse History: From Ad-Hoc to Systematic



Carnegie Mellon
Software Engineering Institute

Today's Talk

Introduction

Product Line Concepts

- What
- Why
- How

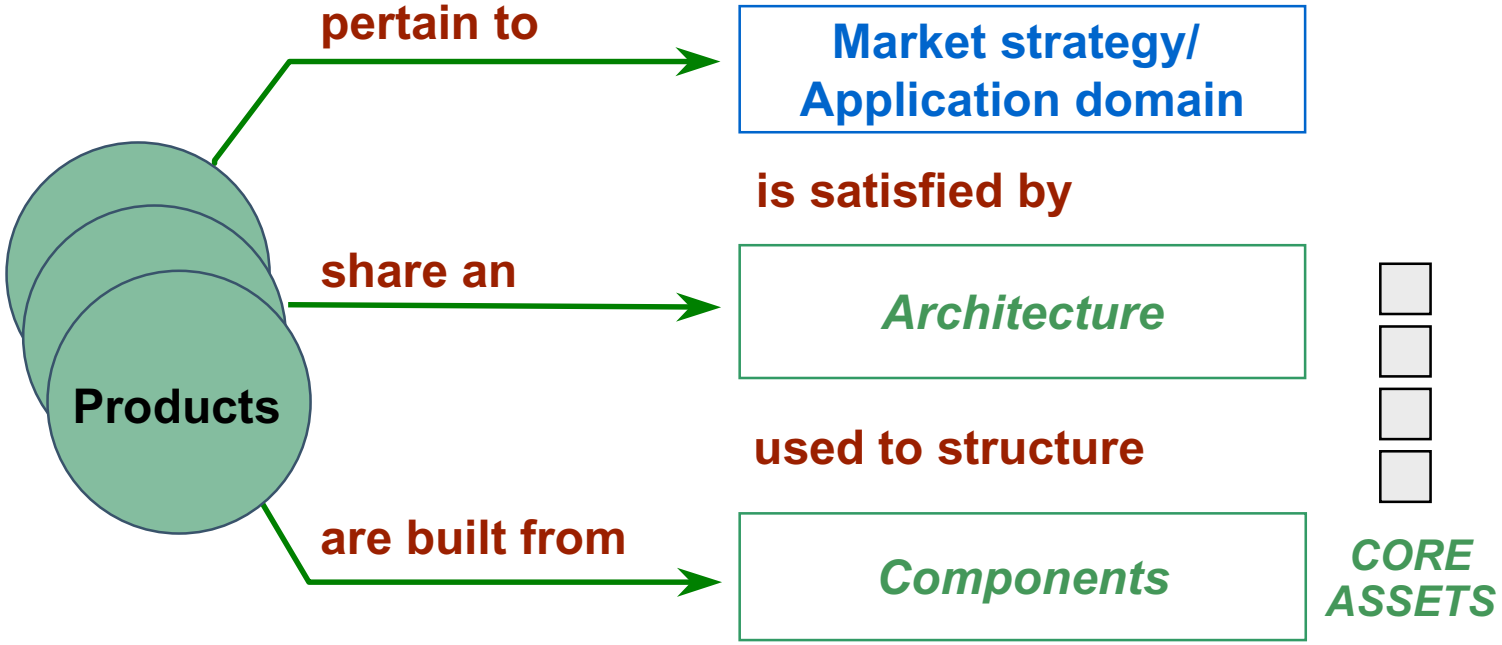
Conclusion

Carnegie Mellon
Software Engineering Institute

What is a Software Product Line?

A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.

Software Product Lines

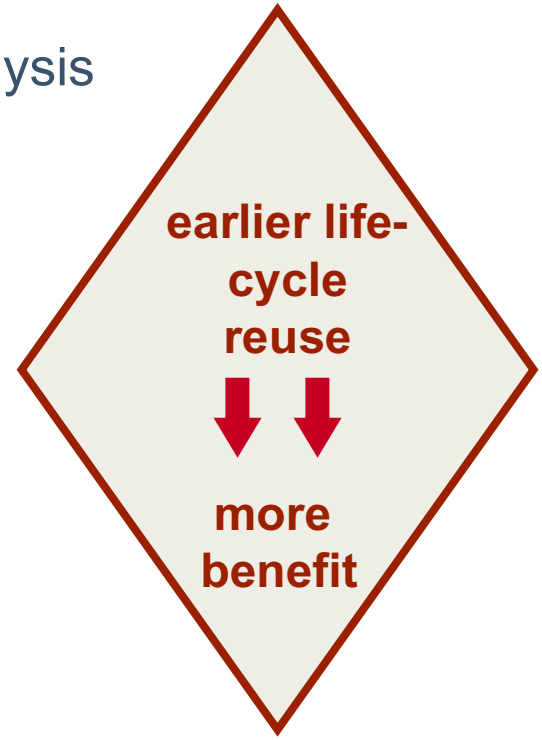


Product lines
- take economic advantage of commonality
- bound variability

How Do Product Lines Help?

Product lines amortize the investment in these and other *core assets*:

- requirements and requirements analysis
- domain model
- software architecture and design
- performance engineering
- documentation
- test plans, test cases, and data
- people: their knowledge and skills
- processes, methods, and tools
- budgets, schedules, and work plans
- components



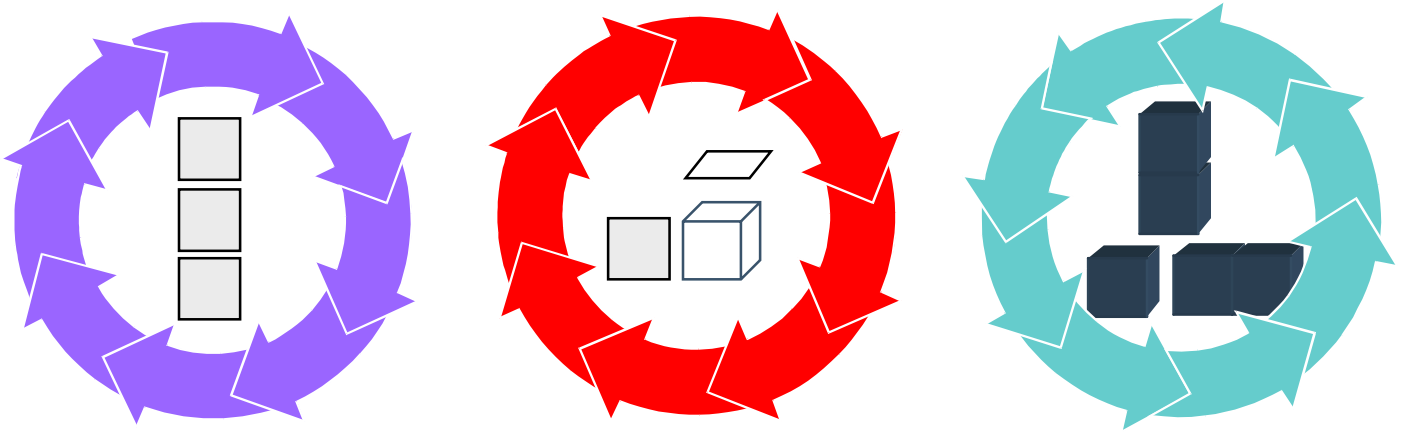
product lines = strategic reuse

The Key Concepts

Use of a common asset base

in production

of a related set of products

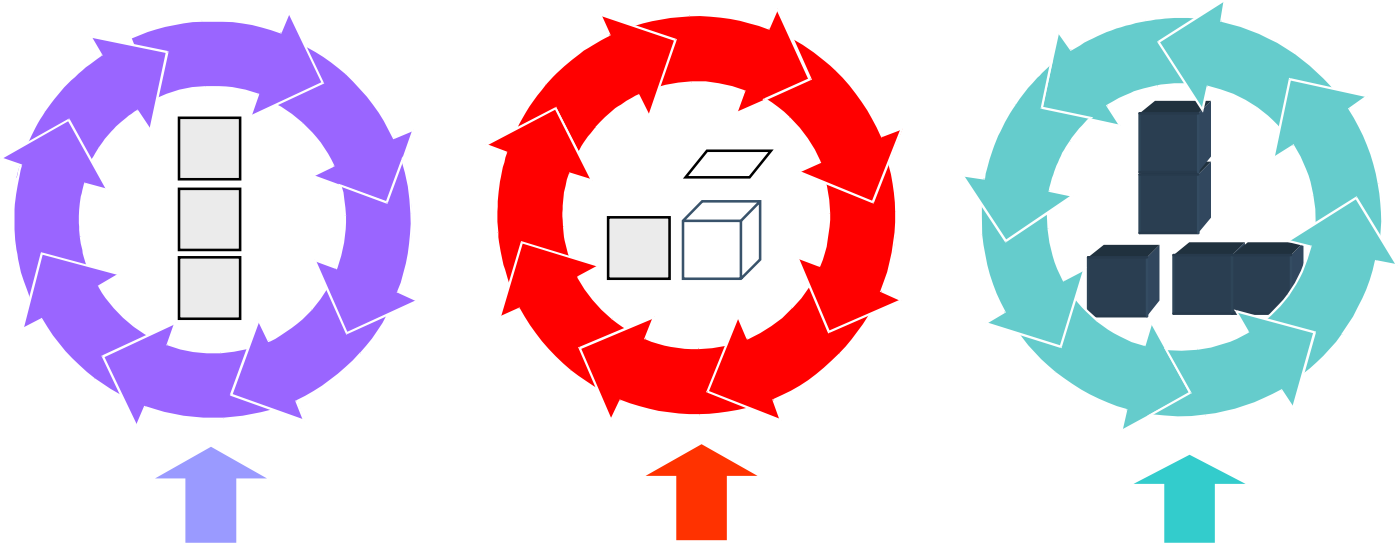


The Key Concepts

Use of a common asset base

in production

of a related set of products



Architecture

Production Plan

Scope Definition
Business Case

Software Product Lines Are Not - 1

Fortuitous Small-Grained Reuse

- Reuse libraries containing algorithms, modules, objects, or components
- Benefits depend on
  - software engineer's predisposition to use what is in the library
  - suitability of library contents for particular needs
  - successful adaptation and integration of library units into the rest of the system
- Reuse is not planned, enabled, or enforced nor are results predictable

Carnegie Mellon
Software Engineering Institute

Software Product Lines Are Not - 2

Single-System Development with Reuse

- Borrowing opportunistically from previous efforts
- Modifying as necessary for the single system only
- Asset base never cultivated

Just Component-Based Development

- Selection of components from an in-house library or the marketplace
- Missing a product line architecture and a production plan as well as management infrastructure

Carnegie Mellon
Software Engineering Institute

Software Product Lines Are Not - 3

Just a Configurable Architecture

- Involves use of a reference architecture or application framework
- Does not involve the planned reuse of other assets

Versions of Single Products

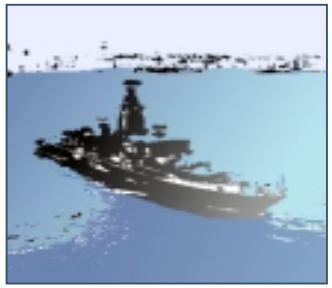
- Involves sequential release of products over time.
- No simultaneous release/support of multiple products

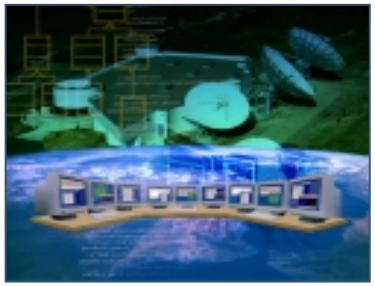
Just a Set of Technical Standards

- Constraints to promote interoperability and to decrease the cost associated with maintenance and support of commercial components
- Does not provide assets and production capability

Product Lines Are

Software product lines involve strategic, planned reuse that yields predictable results.





Carnegie Mellon
Software Engineering Institute

Commercial Examples

Successful software product lines have been built for families of

- Mobile phones
- Command and control ship systems
- Ground-based spacecraft systems
- Avionics systems
- Pagers
- Engine control systems
- Billing systems
- Web-based retail systems
- Printers
- Consumer electronic products
- Acquisition management enterprise systems

Carnegie Mellon
Software Engineering Institute

Today's Talk

Introduction

Product Line Concepts

- What
- Why
- How

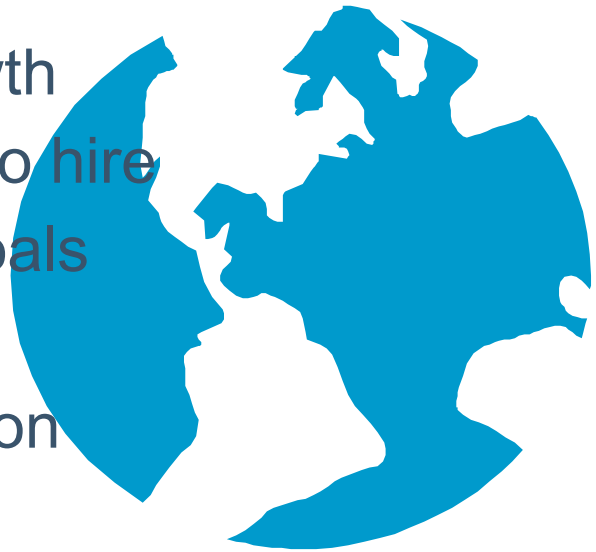
Conclusion

Carnegie Mellon
Software Engineering Institute

Real World Motivation

Organizations use product line practices to:

- achieve large scale productivity gains
- improve time to market
- maintain market presence
- sustain unprecedented growth
- compensate for an inability to hire
- achieve systematic reuse goals
- improve product quality
- increase customer satisfaction
- enable mass customization
- get control of diverse product configurations



Carnegie Mellon
Software Engineering Institute

Organizational Benefits

Improved productivity
by as much as 10x

Decreased time to market (to field, to launch...)
by as much as 10x

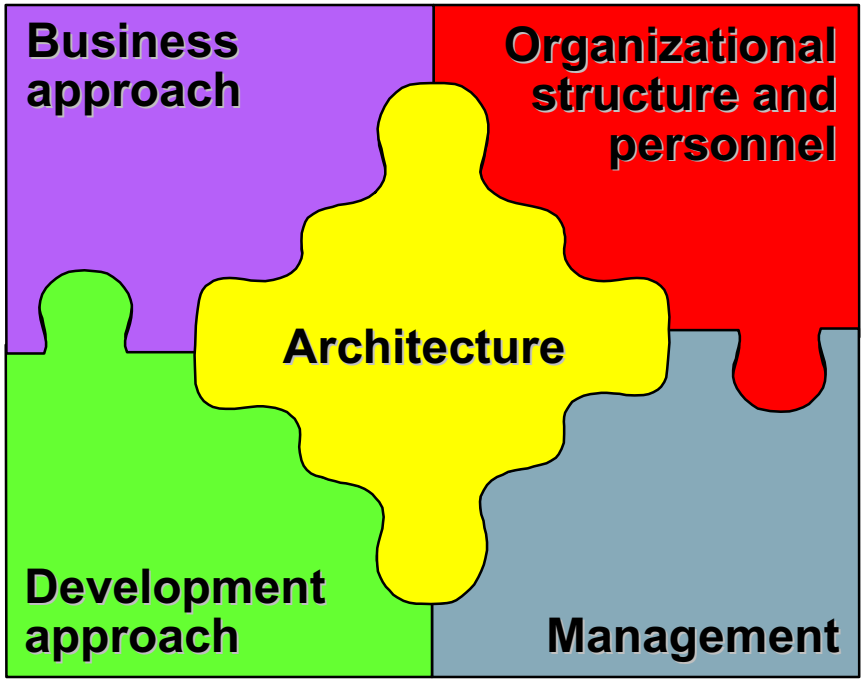
Decreased cost
by as much as 60%

Decreased labor needs
by as much as 10X fewer software developers

Increased quality
by as much as 10X fewer defects

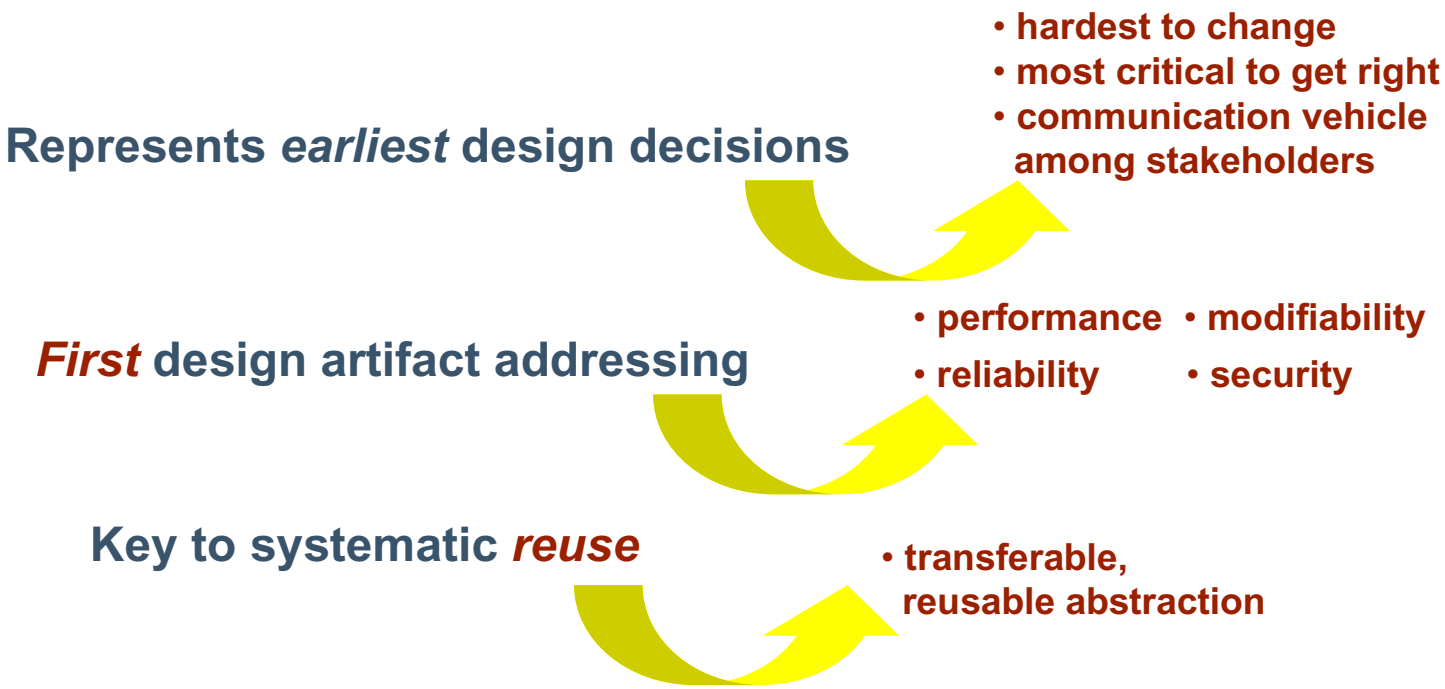
Carnegie Mellon
Software Engineering Institute

Necessary Changes



The architecture is the foundation of everything.

Importance of Architecture



The right architecture paves the way for system success.

The wrong architecture usually spells some form of disaster.

Carnegie Mellon
Software Engineering Institute

Product Line Practice

Contexts for product lines vary widely

- nature of products
- nature of market or mission
- business goals
- organizational infrastructure
- workforce distribution
- process discipline
- artifact maturity

But there are universal essential activities and practices.

Carnegie Mellon
Software Engineering Institute

A Framework for Software Product Line Practice

The three essential activities and the descriptions of the product line practice areas form a conceptual framework for software product line practice.

This framework is evolving based on the experience and information provided by the community.

Version 4.0 – in *Software Product Lines: Practices and Patterns*

Version 4.1 – http://www.sei.cmu.edu/plp/framework.html

Carnegie Mellon
Software Engineering Institute

SEI Information Sources

Case studies, experience reports, and surveys

Workshops, and conferences

Applied research

Collaborations with customers on actual product lines

Carnegie Mellon
Software Engineering Institute

Today's Talk

Introduction

Product Line Concepts

- What
- Why
- How

Conclusion

Product Line Development

The Nature of the Essential Activities

All three activities are interrelated and highly iterative.

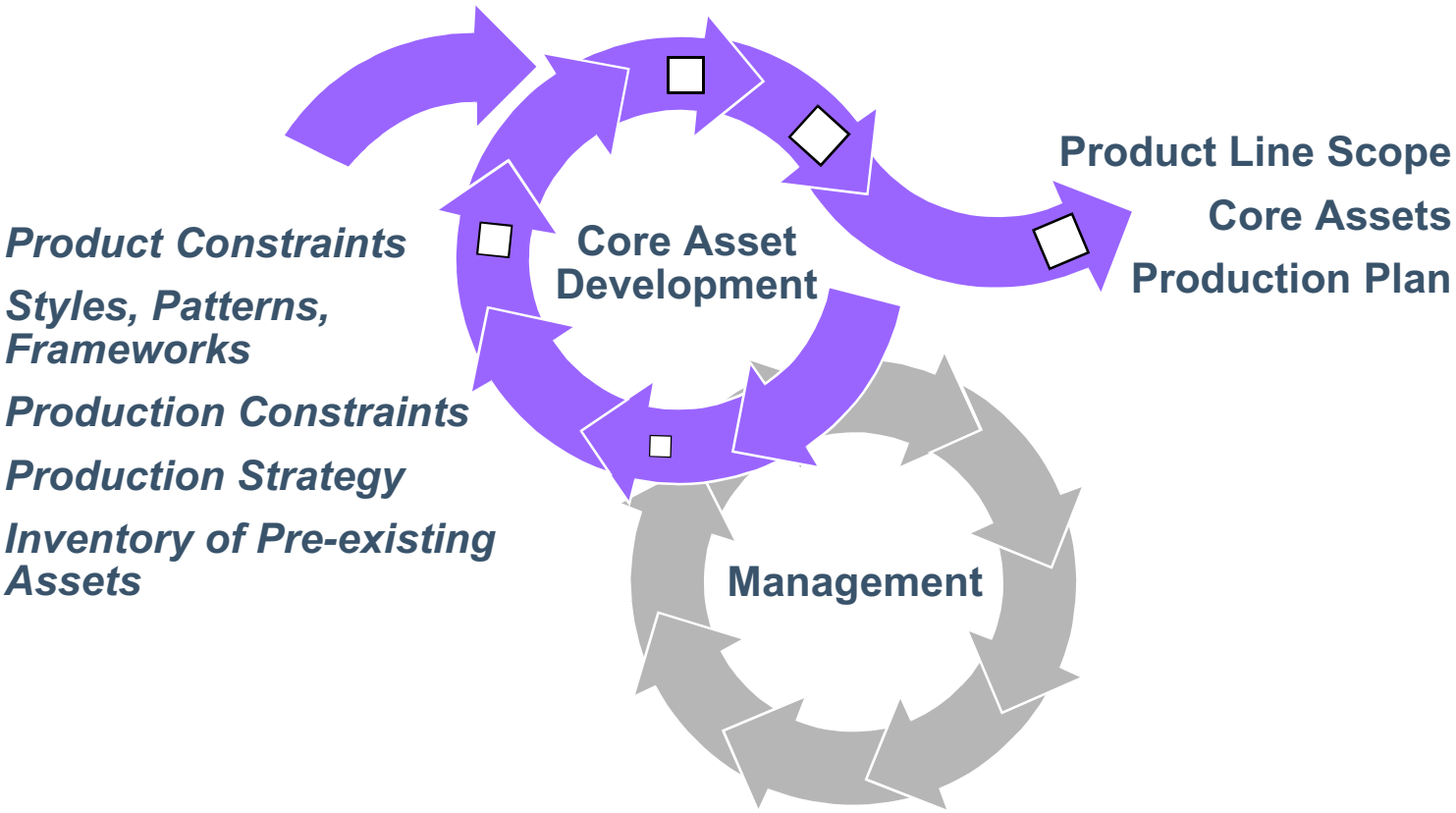
There is no “first” activity.

- In some contexts, existing products are mined for core assets.
- In others, core assets may be developed or procured for future use.

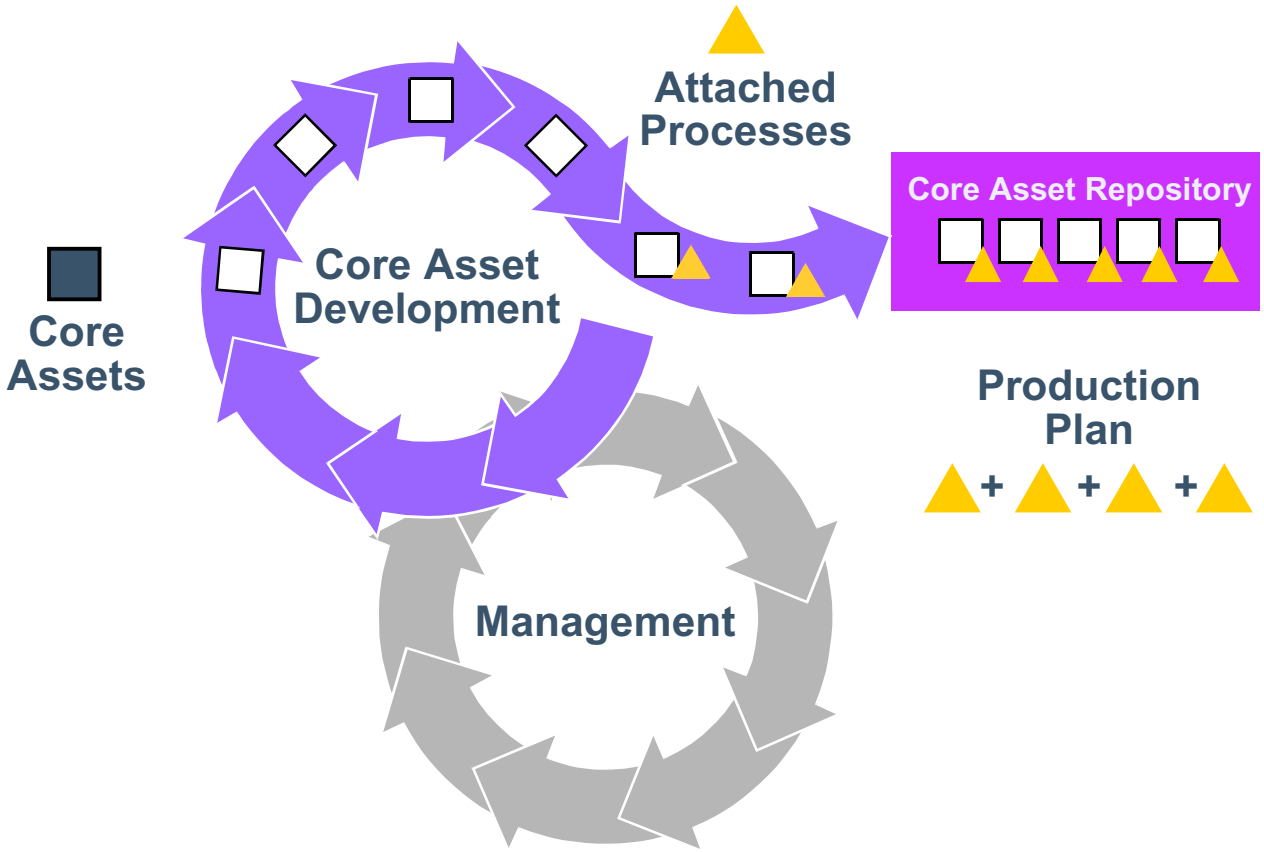
There is a strong feedback loop between the core assets and the products.

Strong management at multiple levels is needed throughout.

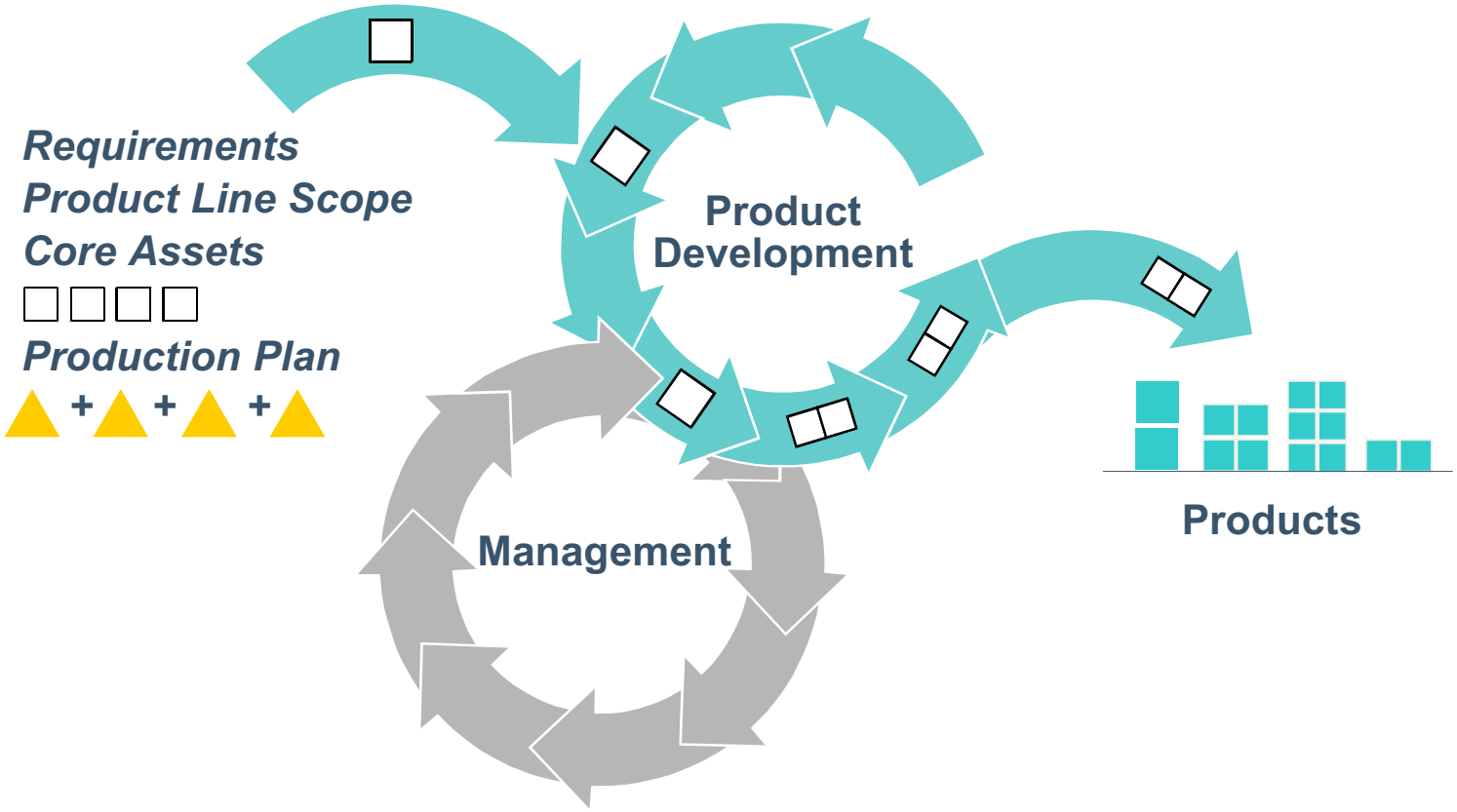
Core Asset Development



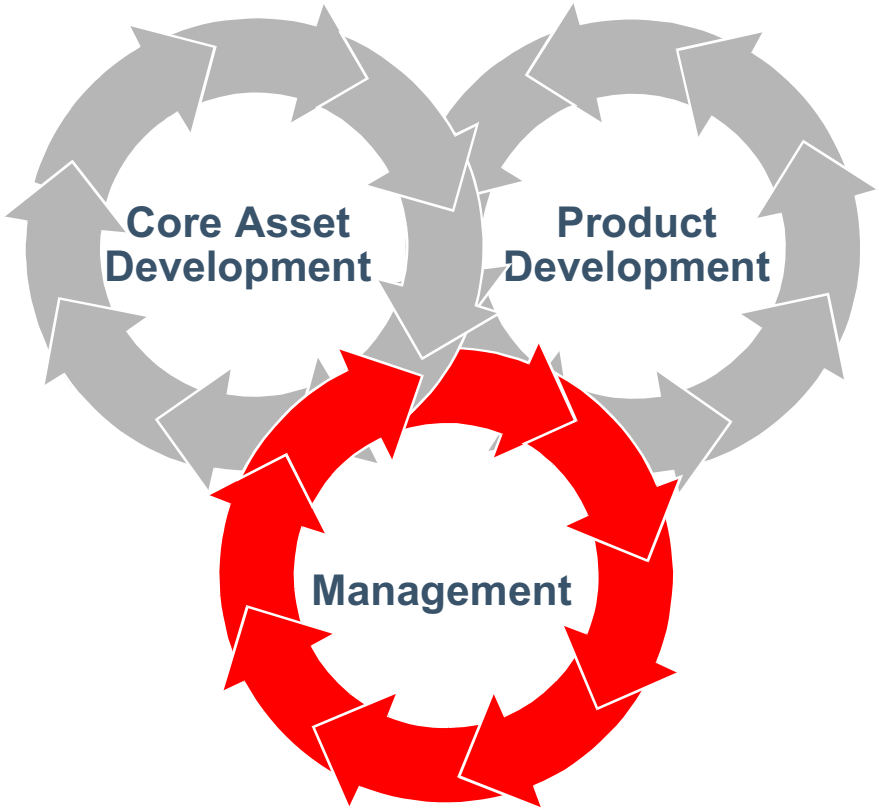
Attached Processes



Product Development



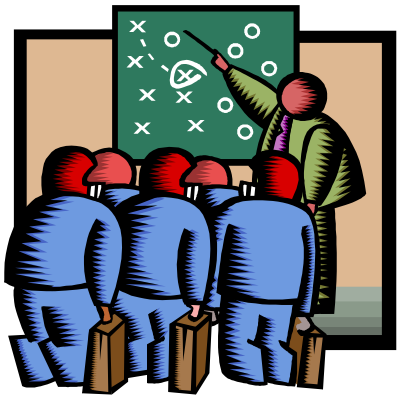
Management



Carnegie Mellon
Software Engineering Institute

Management

Management at multiple levels plays a critical role in the successful product line practice by



- achieving the right organizational structure
- allocating resources
- coordinating and supervising
- providing training
- rewarding employees appropriately
- developing and communicating an acquisition strategy
- managing external interfaces
- creating and implementing a product line adoption plan

Carnegie Mellon
Software Engineering Institute

Managing a Software Product Line Requires Leadership

A key role for a software product line manager is that of champion.

The champion must

- set and maintain the vision
- ensure appropriate goals and measures are in place
- “sell” the product line up and down the chain
- sustain morale
- deflect potential derailments
- solicit feedback and continuously improve the approach

Essential Product Line Activities

Each of these is essential, as is the blending of all three.

Carnegie Mellon
Software Engineering Institute

Different Approaches - 1

Proactive: Develop the core assets first

- Develop the scope first and use it as a "mission" statement.
- Products come to market quickly with minimum code-writing.
- Requires upfront investment and predictive knowledge.

Reactive: Start with one or more products

- From these generate the product line core assets and then future products; the scope evolves more dramatically.
- Much lower cost of entry
- Architecture and other core assets must be robust, extensible, and appropriate to future product line needs

Different Approaches - 2

Incremental: Develop in stages with the plan from the beginning to develop a product line.

- Develop part of the core asset base, including the architecture and some of the components.
- Develop one or more products.
- Develop part of the rest of the core asset base.
- Develop more products.
- Evolve more of the core asset base.
- …..

Carnegie Mellon
Software Engineering Institute

Alternate Terminology

Our Terminology		Alternate Terminology
Product Line	→	Product Family
Core Assets	→	Platform
Business Unit	→	Product Line
Product	→	Customization
Core Asset Development	→	Domain Engineering
Product Development	→	Application Engineering

Carnegie Mellon
Software Engineering Institute

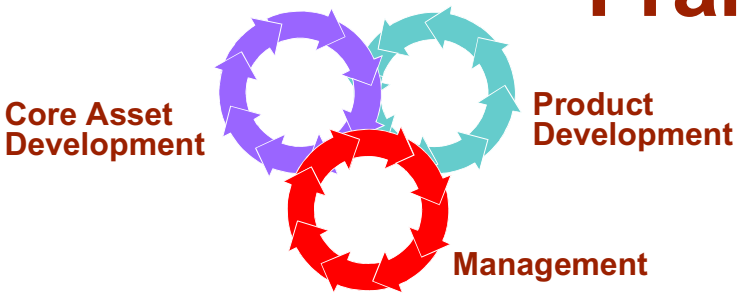
Driving the Essential Activities

Beneath the level of the essential activities are essential practices that fall into practice areas.

A practice area is a body of work or a collection of activities that an organization must master to successfully carry out the essential work of a product line.

Carnegie Mellon
Software Engineering Institute

Framework

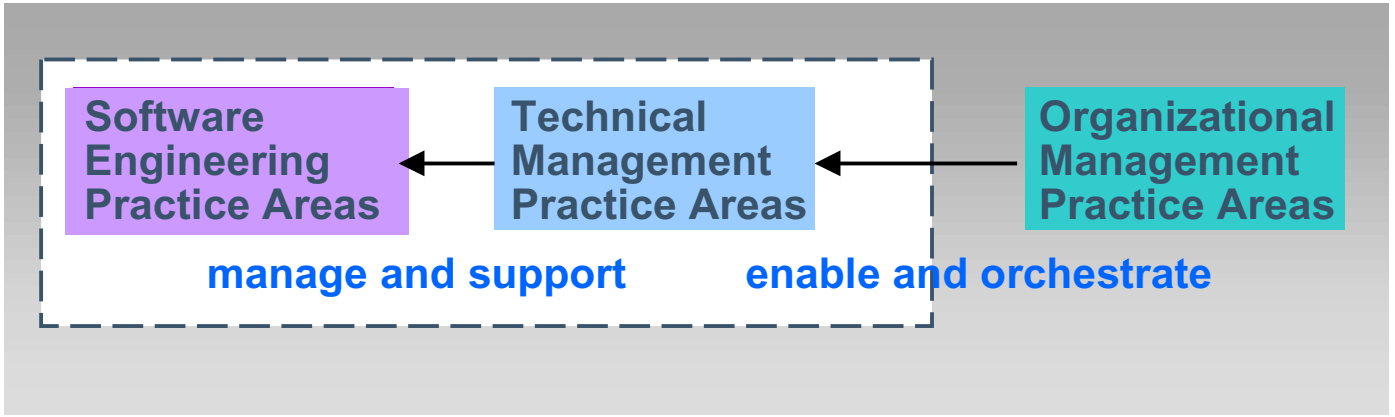


Essential Activities

Software Engineering	Technical Management	Organizational Management
Architecture Definition	Configuration Management	Building a Business Case
Architecture Evaluation	Data Collection, Metrics, and Tracking	Customer Interface Management
Component Development	Make/Buy/Mine/Commission Analysis	Implementing an Acquisition Strategy
COTS Utilization	Process Definition	Funding
Mining Existing Assets	Scoping	Launching and Institutionalizing
Requirements Engineering	Technical Planning	Market Analysis
Software System Integration	Technical Risk Management	Operations
Testing	Tool Support	Organizational Planning
Understanding Relevant Domains		Organizational Risk Management
		Structuring the Organization
		Technology Forecasting
		Training

Practice Areas

Relationships among Categories of Practice Areas



Carnegie Mellon
Software Engineering Institute

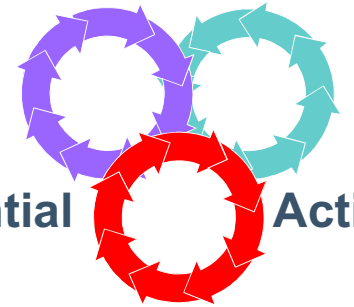
Dilemma: How Do You Apply the 29 Practice Areas?

Organizations still have to figure out how to put the practice areas into play.

29 is a “big” number.

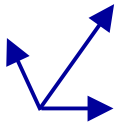
Carnegie Mellon
Software Engineering Institute

Help to Make It Happen

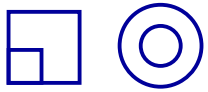


Practice Areas

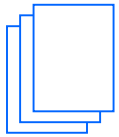
Software Engineering	Technical Management	Organizational Management



Probe



Patterns



Case Studies

Carnegie Mellon
Software Engineering Institute

Case Studies

CelsiusTech – *CMU/SEI-96-TR-016*
http://www.sei.cmu.edu/pub/documents/96.reports/pdf/tr016.96.pdf

Cummins, Inc. *(Software Product Lines: Practices and Patterns)*

Market Maker *(Software Product Lines: Practices and Patterns)*

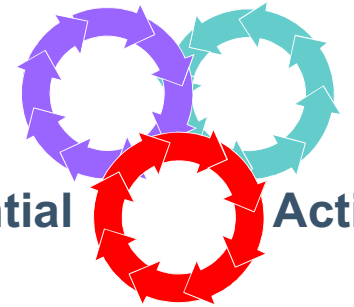
NRO/Raytheon – *CMU/SEI-2001-TR-030*
http://www.sei.cmu.edu/pub/documents/01.reports/pdf/01tr030.pdf

NUWC – *CMU/SEI-2002-TN-018*
http://www.sei.cmu.edu/pub/documents/02.reports/pdf/02tn018.pdf

Salion, Inc. – *CMU/SEI-2002-TR-038*
http://www.sei.cmu.edu/pub/documents/02.reports/pdf/02tr038.pdf

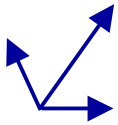
Carnegie Mellon
Software Engineering Institute

Help to Make It Happen

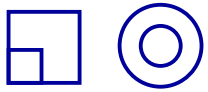


Practice Areas

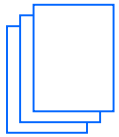
Software Engineering	Technical Management	Organizational Management



Probe



Patterns



Case Studies

Carnegie Mellon
Software Engineering Institute

Patterns Can Help

Patterns are a way of expressing common context and problem-solution pairs.

Patterns have been found to be useful in building architecture, economics, software architecture, software design, software implementation, process improvement, and others.

Patterns assist in effecting a divide and conquer approach.

Carnegie Mellon
Software Engineering Institute

Software Product Line Practice Pattern

Pattern

- **Context** – organizational situation
- **Problem** – what part of a product line effort needs to be accomplished
- **Solution**
  - grouping of practice areas
  - relations among these practice areas (and/or groups if there is more than one)

What to Build Pattern - 1

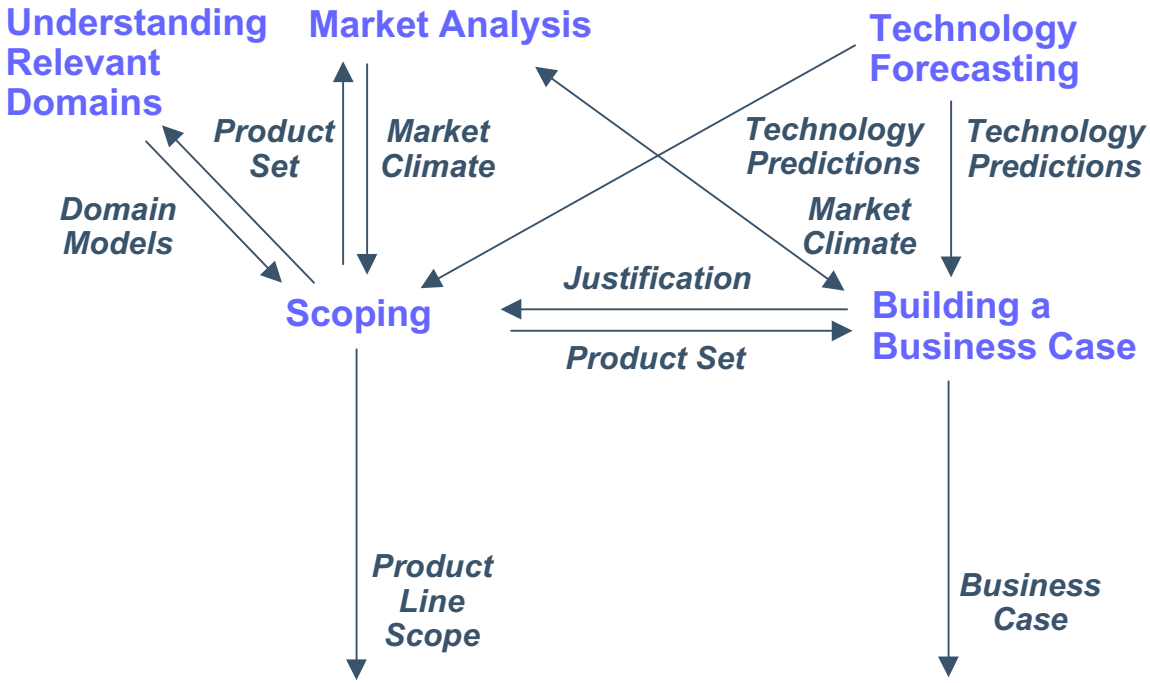
Name:

The *What to Build* pattern helps an organization determine what products ought to be in its software product line – what products to build.

Context:

An organization has decided to field a software product line and knows the general product area for the set of products.

What to Build Pattern - 2



Dynamic Structure

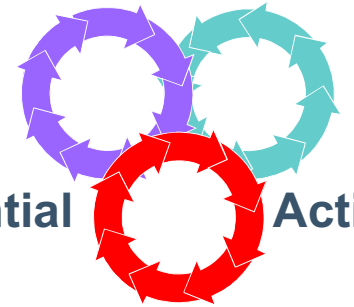
Carnegie Mellon
Software Engineering Institute

Current Set of Patterns

Pattern	Variants
Assembly Line	
Cold Start	Warm Start
Curriculum	
Each Asset	Each Asset Apprentice Evolve Each Asset
Essentials Coverage	
Factory	
In Motion	
Monitor	
Process	Process Improvement
Product Builder	Product Gen
Product Parts	Green Field Barren Field Plowed Field
What to Build	Analysis Forced March

Carnegie Mellon
Software Engineering Institute

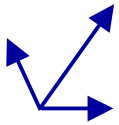
Help to Make It Happen



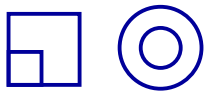
Essential Activities

Practice Areas

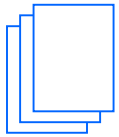
Software Engineering	Technical Management	Organizational Management



Probe

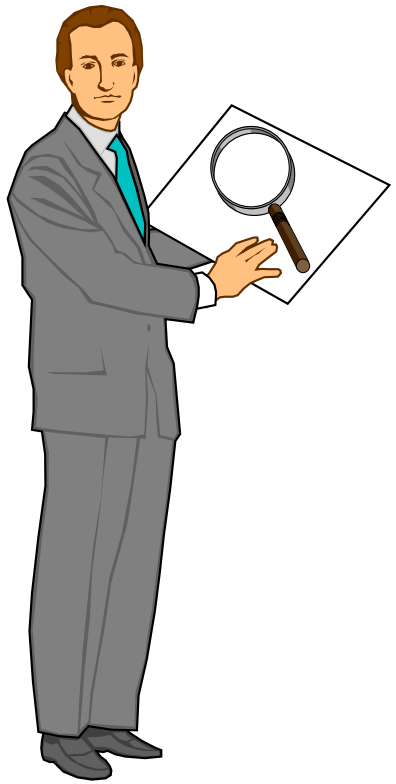


Patterns



Case Studies

What is a Product Line Technical Probe?



A method for examining an organization's readiness to adopt or ability to succeed with a software product line approach

- diagnostic tool based on the Framework for Software Product Line Practice
- practice areas are used in the data collection and analysis

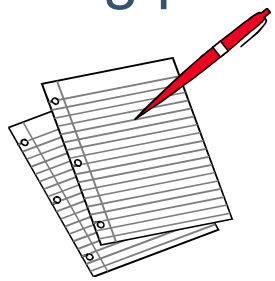
Probe Outcomes

Set of findings that portray organizational

- strengths
- challenges

with regard to a product line approach.

Findings can be used to develop an action plan with the goal of making the organization more capable of achieving product line success.



Carnegie Mellon
Software Engineering Institute

Today's Talk

Introduction

Product Line Concepts

- What
- Why
- How

Conclusion

Carnegie Mellon
Software Engineering Institute

In a Nutshell

Software product lines epitomize the concept of strategic, planned reuse.

The product line concept is about more than a new technology. It is a new way of doing one's software business.

There are essential product line activities and practices areas as well as product line patterns to make the move to product lines more manageable.

Carnegie Mellon
Software Engineering Institute

What's Different About Reuse with Software Product Lines?

Business dimension

Iteration

Architecture focus

Pre-planning

Process and product connection

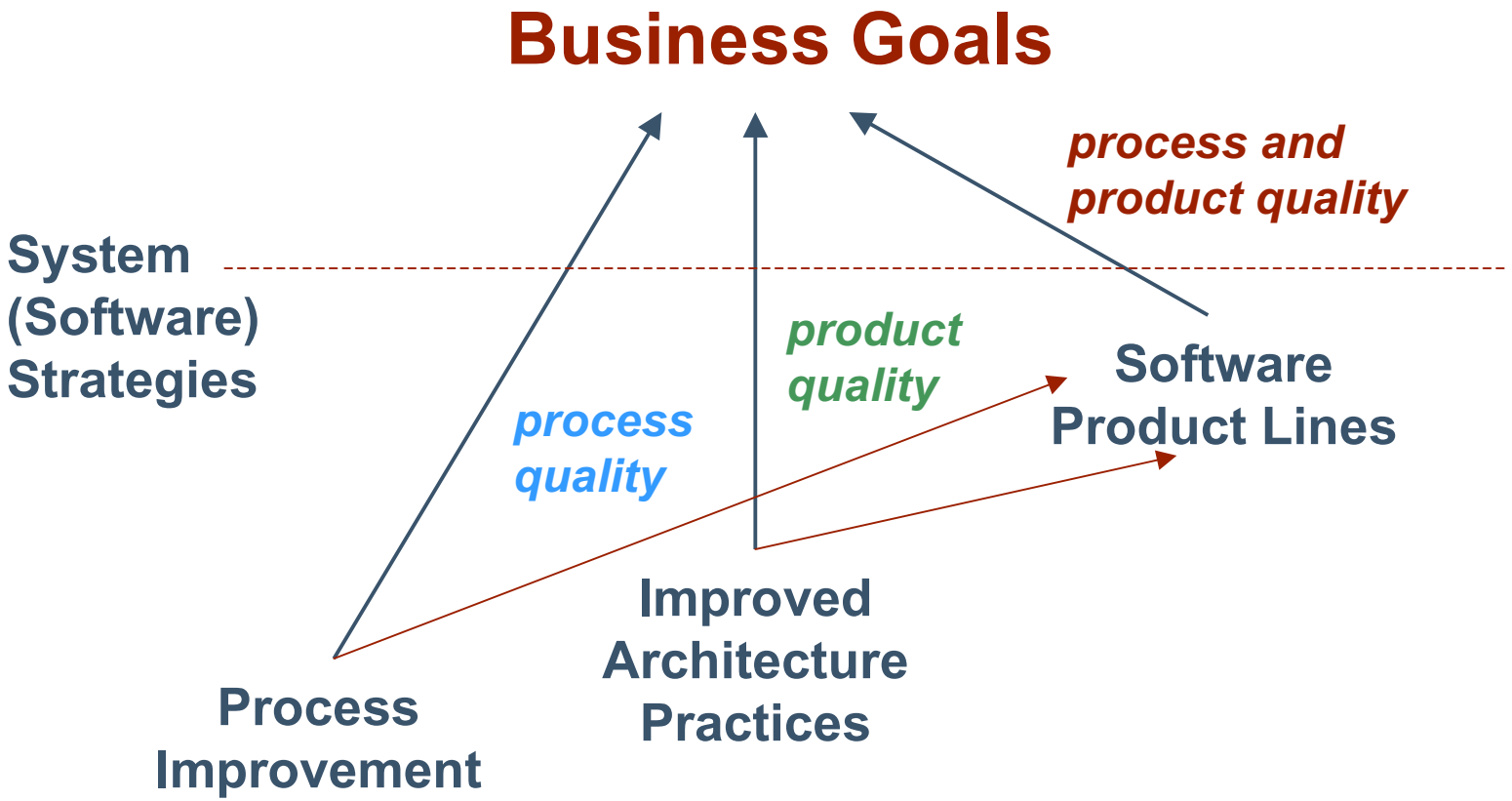
Carnegie Mellon
Software Engineering Institute

Based on Our Experience

- Product line business practices cannot be effected without management commitment and involvement.
- Organization size doesn't matter.
- Organizational culture plays a major role in adoption success.
- Organizations need support: guidance, diagnostics, methods, and tools.
- The lack of an architecture focus and/or talent can kill an otherwise promising product line effort.
- Process discipline is critical.
- The community needs more quantitative data to support product line adoption.
- The cultural barriers and cost of adoption are major impediments to widespread transition.
- Software product line practice is at the "chasm." (in Geoffrey Moore's terms: *Crossing the Chasm)*

Software Product Line Strategy in Context

Software Product Line Strategy in Context



Carnegie Mellon
Software Engineering Institute

The Time is Right

Rapidly maturing, increasingly sophisticated software development technologies including *object technology, component technology, standardization of commercial middleware.*

A global realization of the *importance of architecture*

A universal recognition of the need for *process discipline.*

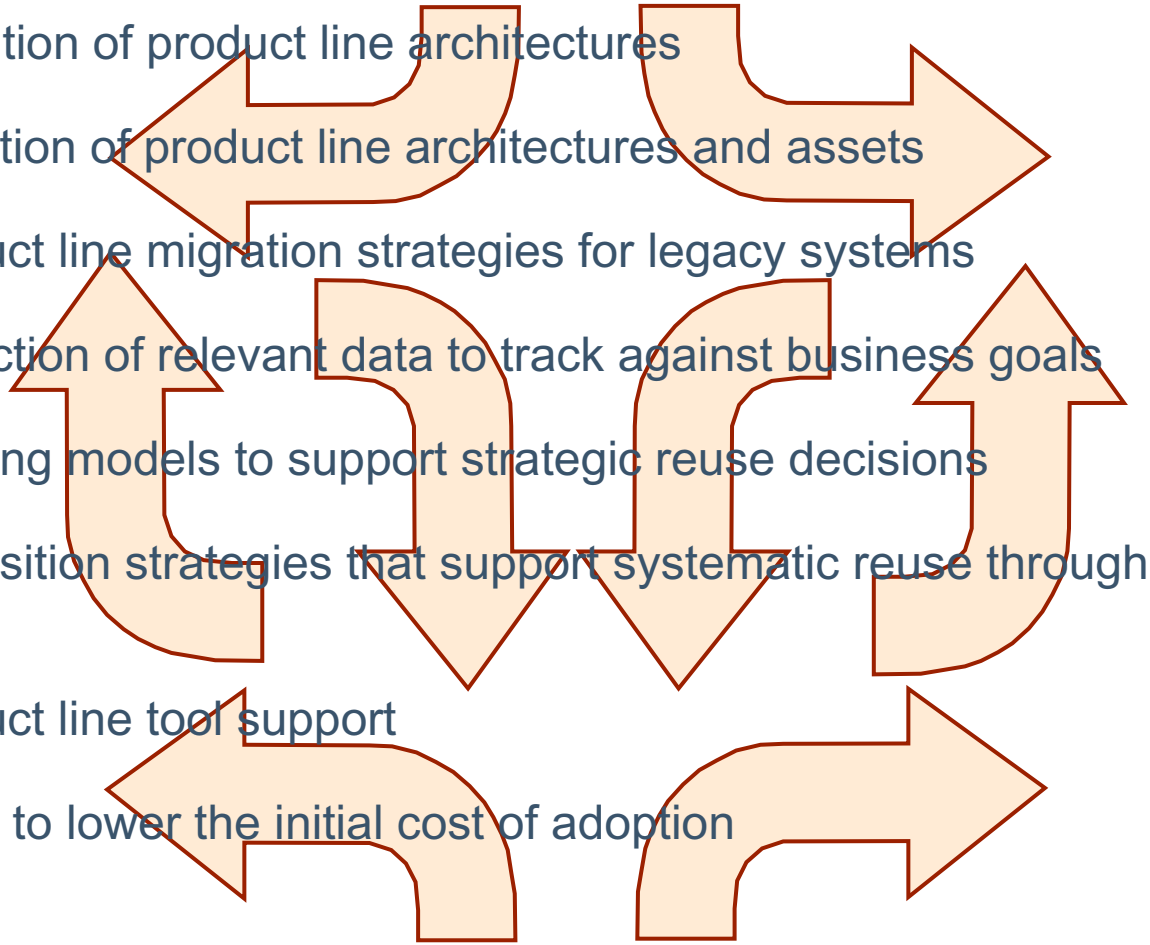
Role models and case studies that are emerging in the literature and trade journals.

Conferences, workshops, and education programs that are now including product lines in the agenda.

Company and inter-company *product line initiatives.*

Rising recognition of the *amazing cost/performance savings* that are possible.

Remaining Challenges



Definition of product line architectures

Evolution of product line architectures and assets

Product line migration strategies for legacy systems

Collection of relevant data to track against business goals

Funding models to support strategic reuse decisions

Acquisition strategies that support systematic reuse through product lines

Product line tool support

Ways to lower the initial cost of adoption

Carnegie Mellon
Software Engineering Institute

Summary of SEI Contributions

Practice Integration:

- A Framework for Software Product Line Practice[SM], Version 4.1, http://www.sei.cmu.edu/plp/framework.html
- Acquisition Companion to the Framework

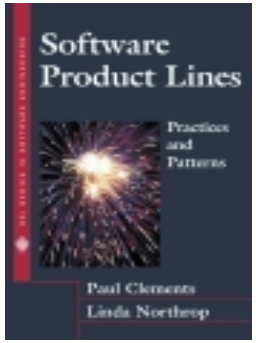
Techniques and Methods

- product line analysis
- architecture definition – Attribute-Driven Design (ADD)
- architecture evaluation – Architecture Tradeoff Analysis Method[SM] (ATAM[SM])
- mining assets – Options Analysis for Reengineering[SM] (OAR[SM])
- Product Line Technical Probe[SM]

Book

Software Product Lines: Practices and Patterns

- Practices (Framework, Version 4.0)
- patterns
- case studies



Conferences

SPLC 2004 – Sept 2004

Carnegie Mellon
Software Engineering Institute

Spreading the Software Product Line Word

Software product line concepts, practices, and patterns

Architecture design

Mining assets

Product line analysis

Acquisition Guidelines

Courses

Essentials of Software Product Lines

Software Product Lines

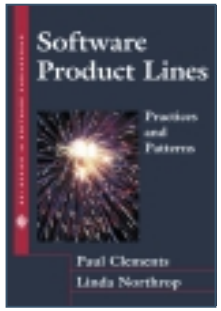
Attribute-Driven Design

Options Analysis for Reengineering[SM]

Product Line Analysis Tutorial

Acquisition Executive Tutorial

Book



Reports

Web

Carnegie Mellon
Software Engineering Institute

Widespread Transition: SEI Software Architecture Curriculum

Six courses

- Software Architecture Familiarization
- Documenting Software Architectures
- Software Architecture Design and Analysis
- Software Product Lines
- ATAM Evaluator Training
- ATAM Facilitator Training

Three certificate programs

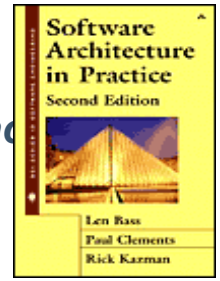
- Software Architecture Professional
- ATAM Evaluator
- ATAM Lead Evaluator

In addition

- Architecture Analysis Guidelines for Acquisition Managers

Carnegie Mellon
Software Engineering Institute

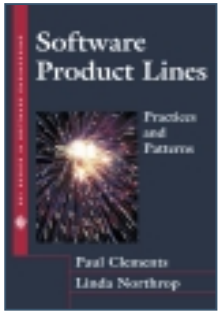
Associated Texts



Software Architecture in Prac
2nd Edition

Documenting Software Architectures: Views and Beyond

Evaluating Software Architectures: Methods and Case Studies



Software Product Lines: Practices and Patterns

Carnegie Mellon
Software Engineering Institute

Ongoing SEI Research

Variability mechanisms
Asset evolution
Production plan definition and implementation
Product line adoption phases and strategies

Architectural tactics
Architect Expert

Predictable assembly from certifiable components (PACC)

Carnegie Mellon
Software Engineering Institute

Final Word

If properly managed, the benefits of a product line approach far exceed the costs.

Strategic software reuse through a well-managed product line approach achieves business goals for:

- efficiency
- time to market
- productivity, and
- quality

Software product lines: Reuse that pays.

Carnegie Mellon
Software Engineering Institute

Questions – Now or Later

Linda Northrop
Director
Product Line Systems Program
Telephone: 412-268-7638
Email: lmn@sei.cmu.edu

U.S. mail:
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

World Wide Web:
http://www.sei.cmu.edu/ata
http://www.sei.cmu.edu/plp

SEI Fax: 412-268-5758

Paul Clements
Product Line Systems Program
Telephone: 512-453-1471
Email: clements@sei.cmu.edu

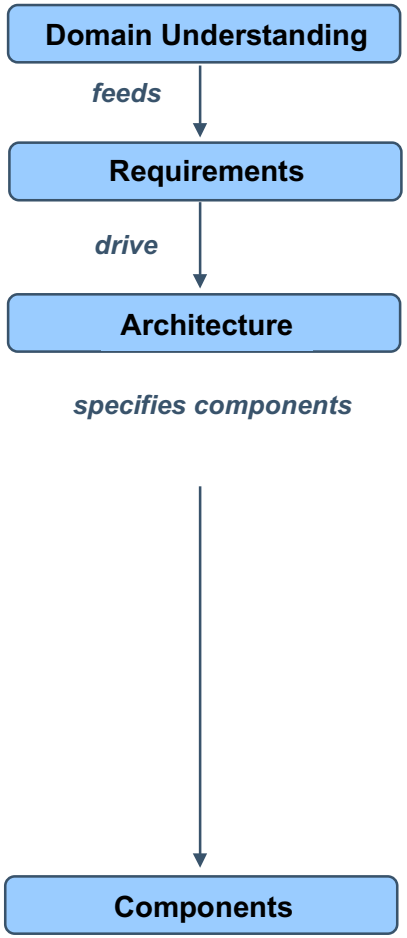
Business Development
Product Line Systems Program
Tim Denmeade
Telephone: 412-268-8243
Email: td@sei.cmu.edu

Carnegie Mellon
Software Engineering Institute

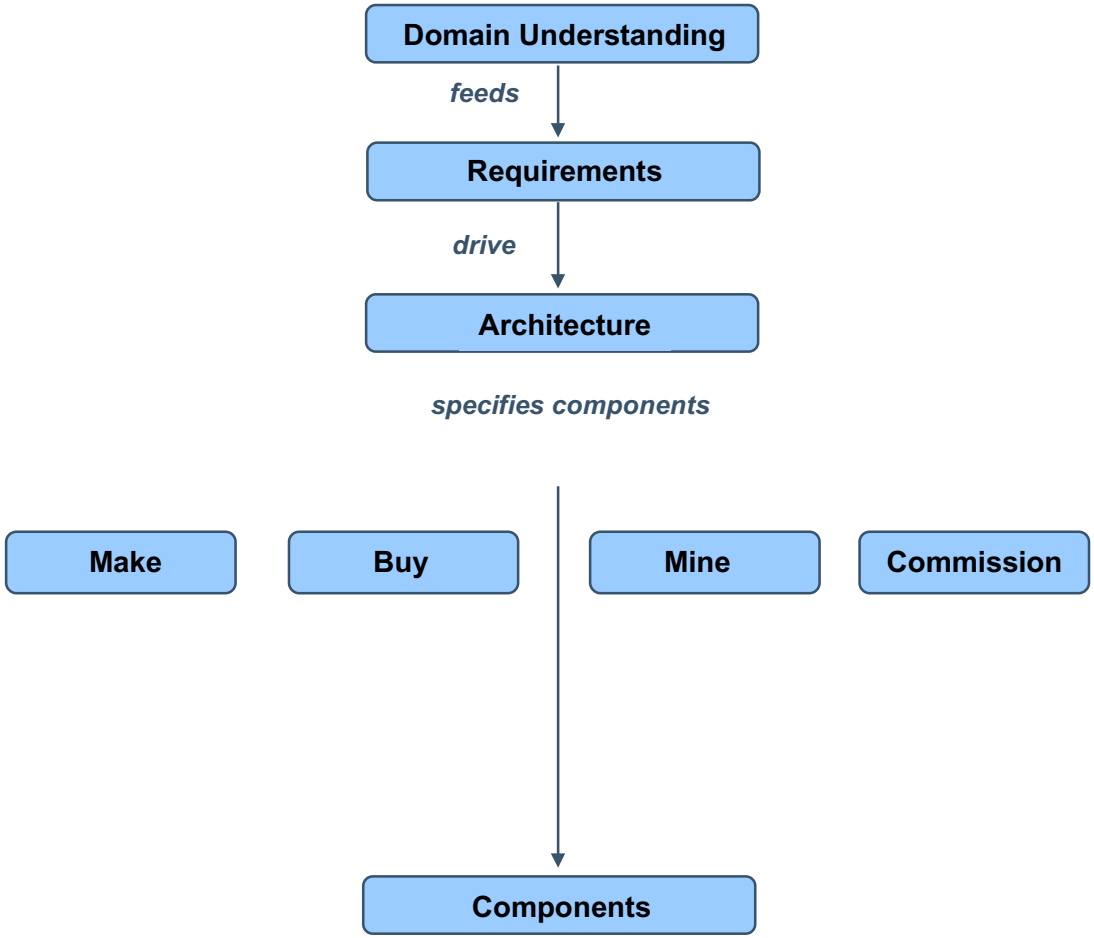
Addendum

Extra slides for further explanation.

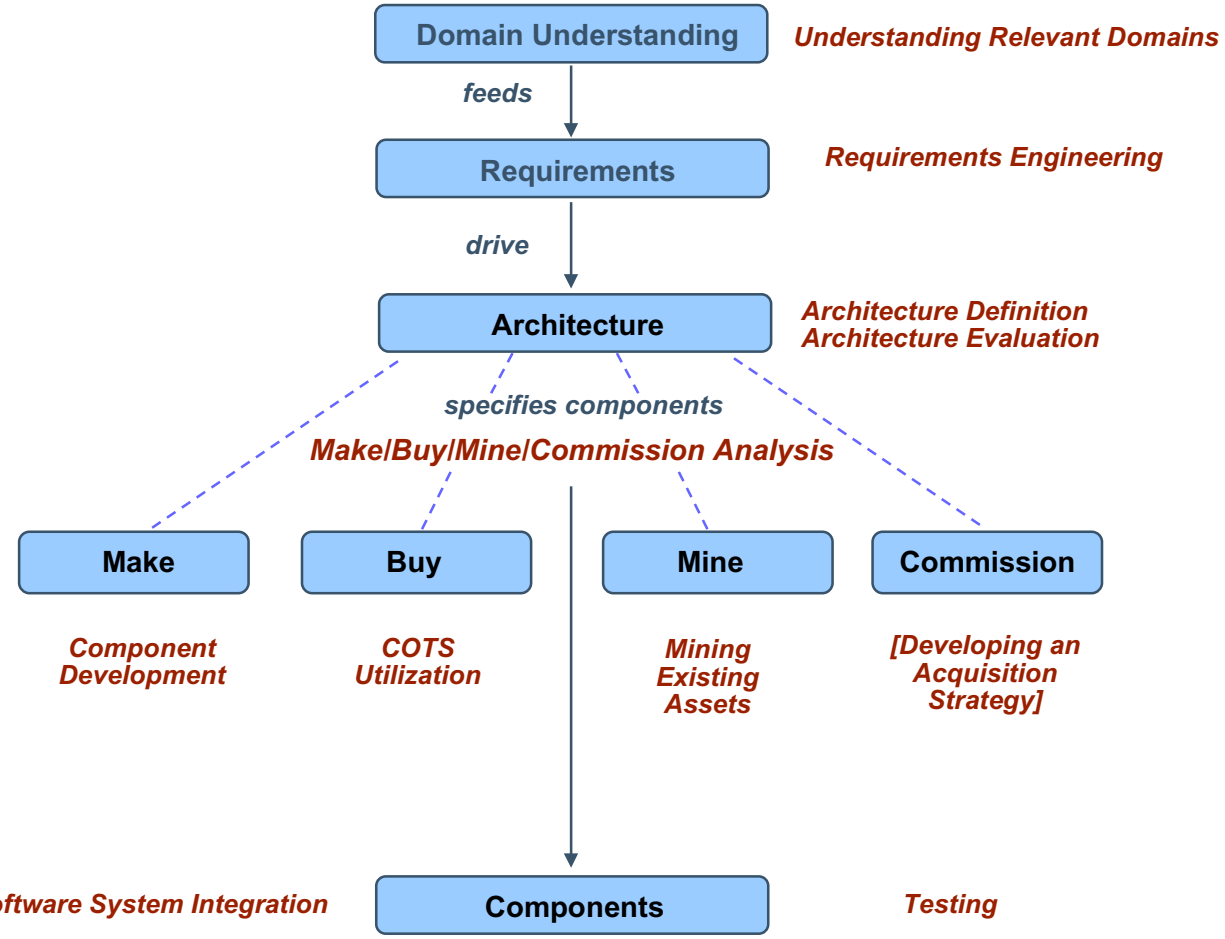
Practice Area Relationships



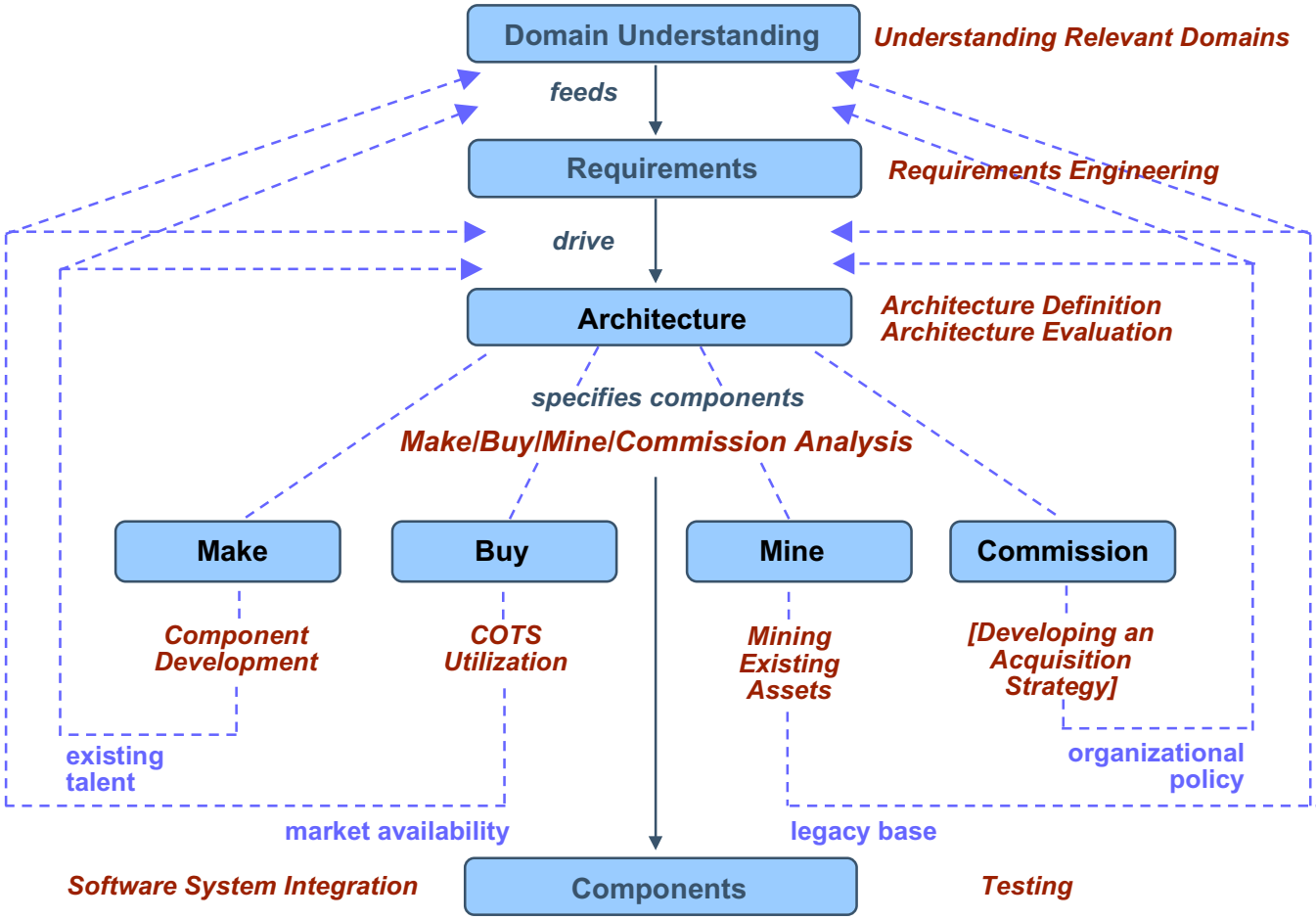
Practice Area Relationships



Practice Area Relationships



Practice Area Relationships



Carnegie Mellon
Software Engineering Institute

Factory Pattern - 1

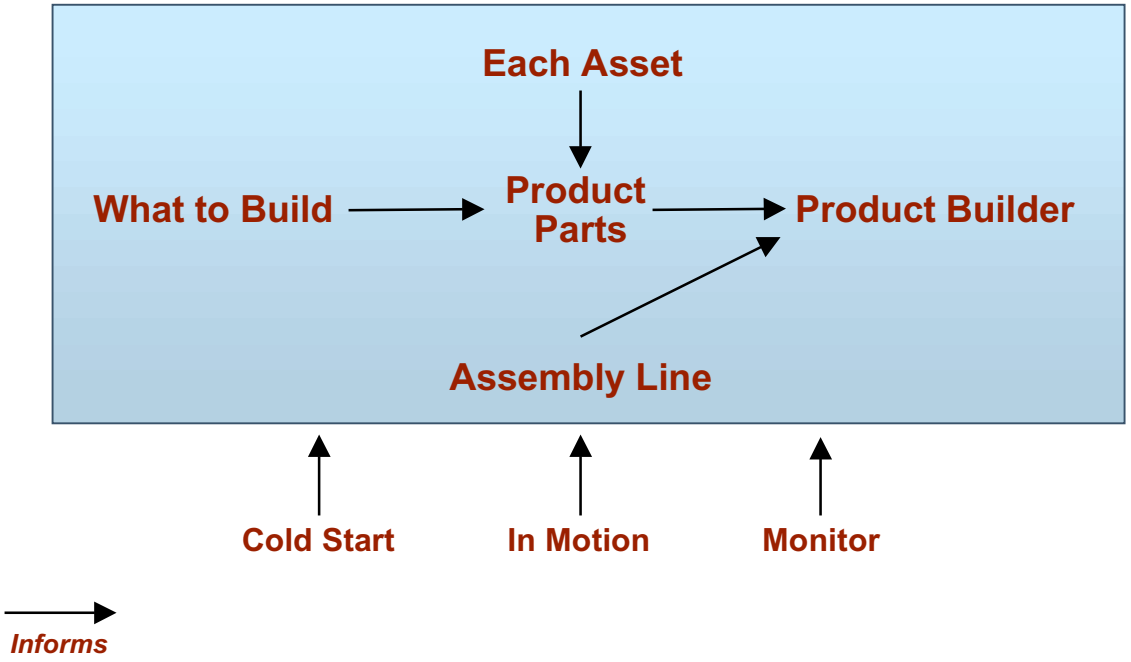
Name:

The Factory is a composite pattern that describes the entire product line organization.

Context:

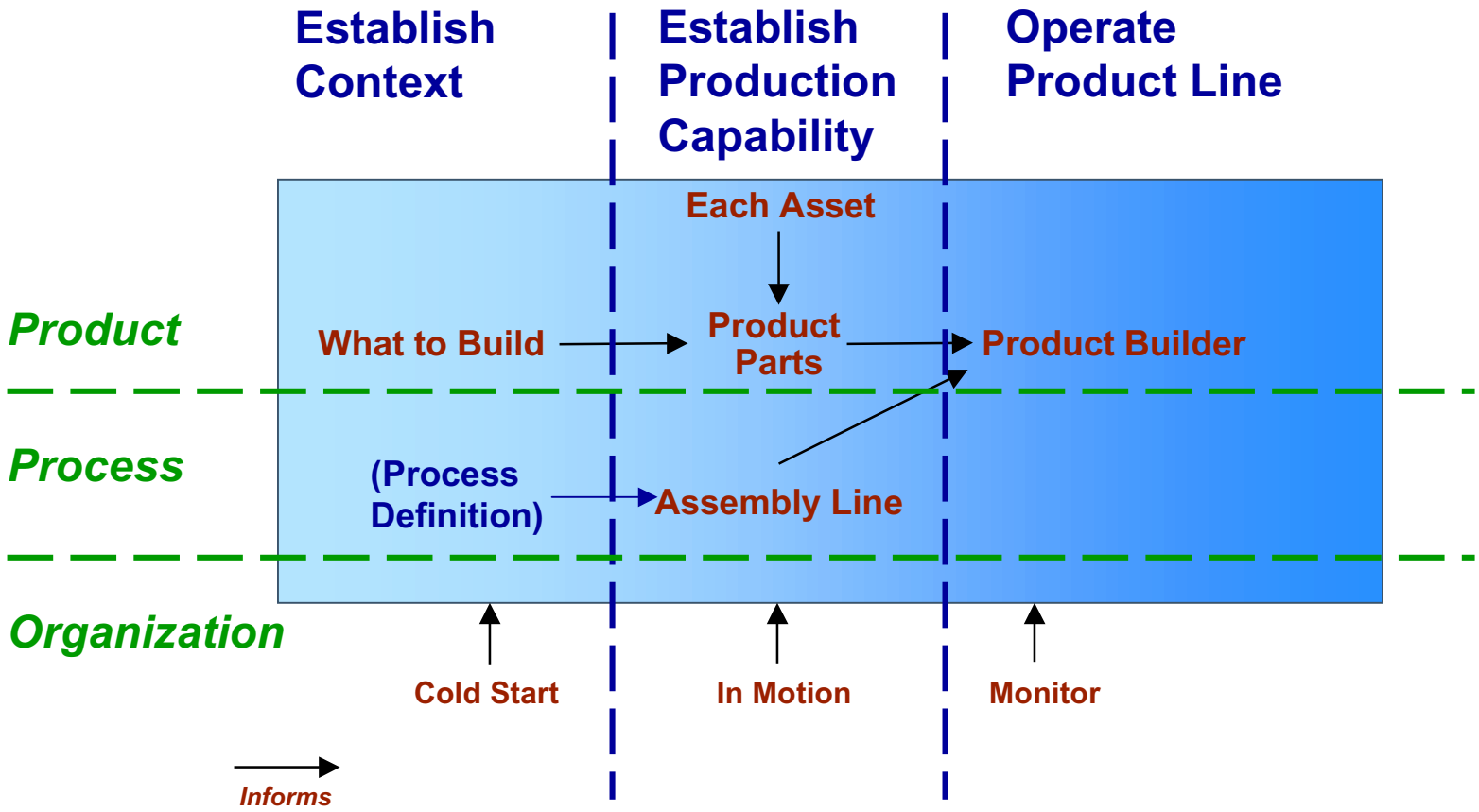
An organization is considering (or fielding) a product line.

Factory Pattern - 2



Dynamic Structure

Phases of Product Line Mastery



Factory Pattern

Carnegie Mellon
Software Engineering Institute

Associated Practice Areas

	Establish Context	Establish Production Capability	Operate Product Line
Product	Marketing Analysis Understanding Relevant Domains Technology Forecasting Building a Business Case Scoping	Requirements Engineering Architecture Definition Architecture Evaluation Mining Existing Assets Component Development COTS Utilization Software System Integration Testing	Requirements Engineering Architecture Definition Architecture Evaluation Mining Existing Assets Component Development COTS Utilization Software System Integration Testing
Process	Process Definition	Configuration Management Tool Support Data Collection, Metrics and Tracking Technical Planning	Configuration Management Tool Support Data Collection, Metrics and Tracking Technical Planning
Organization	Launching and Institutionalizing Funding Structuring the Organization Operations Organizational Planning Customer Interface Management Organizational Risk Management Developing an Acquisition Strategy Training	Launching and Institutionalizing Funding Structuring the Organization Operations Organizational Planning Customer Interface Management Organizational Risk Management Developing an Acquisition Strategy Training	Data Collection, Metrics and Tracking Technical Risk Management

Software System Development

Functional Software Requirements

If function were all that mattered, any monolithic software would do, ..***but other things matter***…

The important quality attributes and their characterizations are key.

- Modifiability
- Interoperability
- Availability
- Security
- Predictability
- Portability
- :

analysis, design, development

Quality Attribute Drivers → Software Architecture → Software

has these qualities

Carnegie Mellon
Software Engineering Institute

Principle 1

Quality requirements can be collected and stated in a fashion meaningful for design

We have focused on six quality attributes:

- availability
- modifiability
- performance
- security
- testability
- usability

Carnegie Mellon
Software Engineering Institute

Principle 2

There are known architectural techniques to achieve quality attributes and these techniques can be enumerated.

For the six quality attributes –availability, modifiability, performance, security, testability, usability - we have enumerated a collection of “tactics”

Def: A *tactic* is an design decision that helps achieve a specific quality attribute response.

Carnegie Mellon
Software Engineering Institute

Tactics for Performance

The tactics for performance are the following:

Performance		
Resource Demand	**Resource Management**	**Resource Arbitration**
Increase Computational Efficiency Reduce Computational Overhead Manage Event Rate Control Frequency of Sampling	Introduce Concurrency Maintain Multiple Copies Increase Available Resources	Scheduling Policy

Tactics vis a vis architectural patterns

Tactics are more general than patterns.

- “increase computational efficiency” is a design decision that is independent of any particular pattern
- “maintain multiple copies” is used in any pattern that has caching as well as in availability patterns

Patterns package tactics - any pattern utilizes a collection of tactics

Carnegie Mellon
Software Engineering Institute

Challenge Motivating PACC

Software components are critical to today's systems and product lines
BUT the behavior of component assemblies is unpredictable.

- "interface" abstractions are not sufficiently descriptive
- behavior of components is, in part, an *a priori* unknown
- behavior of component assemblies must be discovered

The result is costly development and decreased assurance.

The net effect is slowed adoption of component technology.

The Vision

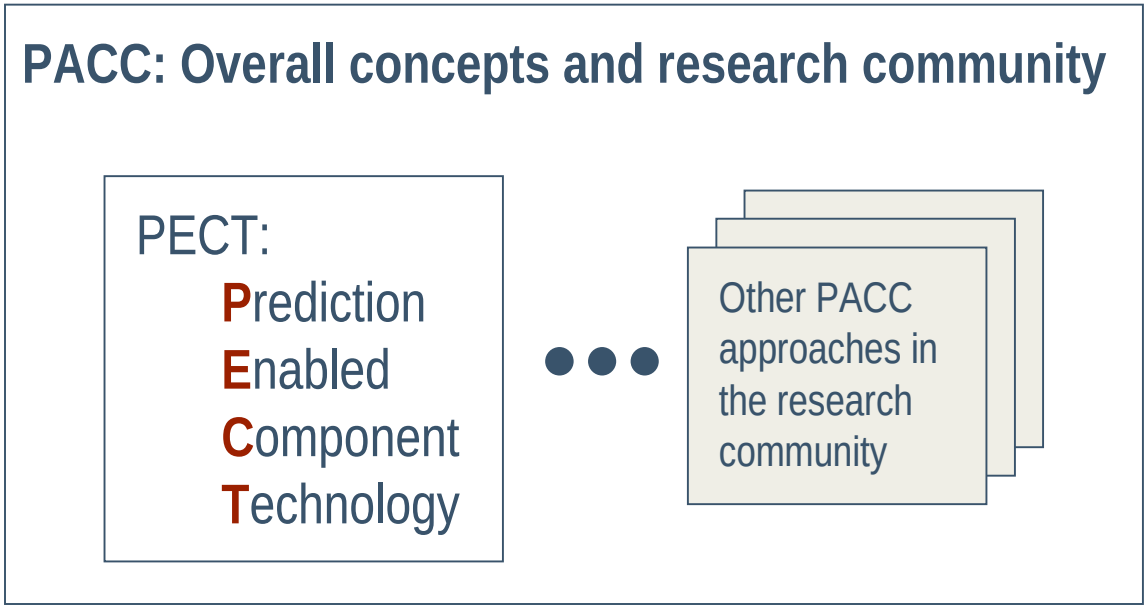
Our vision is to provide the engineering methods and technologies that will enable

- properties of assemblies of components to be reliably predicted, by construction
- properties of components used in predictions to be objectively trusted

We refer to the end-state as having achieved predictable assembly from certifiable components (PACC)

Carnegie Mellon
Software Engineering Institute

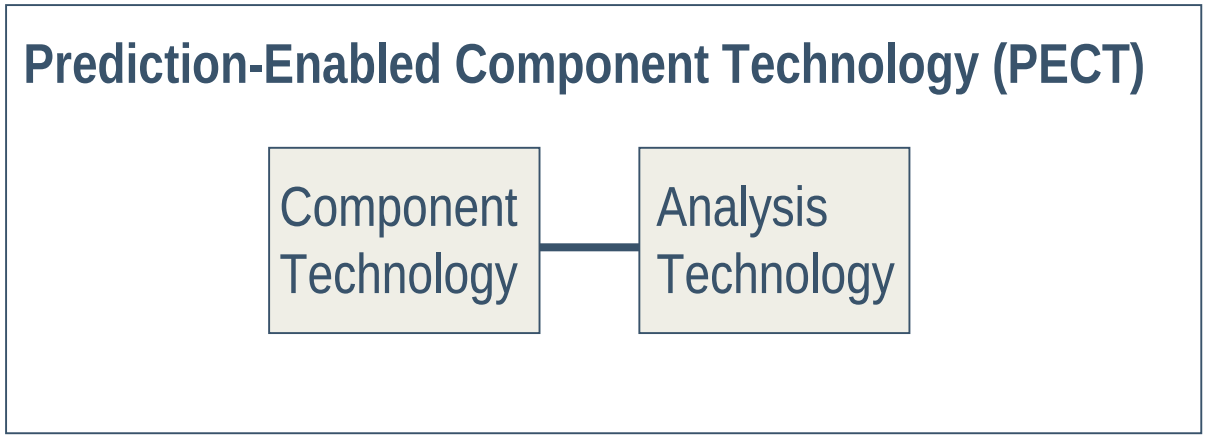
Prediction Enabled Component Technology (PECT)



PECT is the approach we propose to achieve PACC goals.

Carnegie Mellon
Software Engineering Institute

PECT Reference Concept



At the grossest level, PECT is the integration of component technology with analysis technology

Carnegie Mellon
Software Engineering Institute

Industrial Demonstration

Customer: ABB Corporate Research Center

Customer Information

- Transforming from heavy industry in power plant equipment to IT products and services in process automation

Purpose

- First year of collaboration to demonstrate the feasibility of PACC in substation automation
- Second year of collaboration to demonstrate the feasibility of PACC in industrial robotics

Problem Being Solved

- Predictable assembly from certifiable components in substation automation domain
  - operator level latency (PECT)
  - controller level latency (PECT)
  - combined operator-controller latency ($PECT^2$)

  and in robotics domain
- Reliability and safety scenarios are under investigation

Status

- Feasibility study for substation automation completed
- Robotics work underway

Carnegie Mellon
Software Engineering Institute

Status

PACC premises were validated on an internal system and through an ABB Feasibility Study.

PACC became an initiative as of October 2002.

The emphasis of work in 2002-03 is to ready PECT for practitioner use

- practical automation for building and using PECTs
  - conceptual framework of PECT was generalized in and was more rigorously defined
  - specification language (CCL) was defined and tools are currently being developed
- model checking was introduced for reliability verification
- technical advances in timing and reliability analysis

paves the way to real industry trial, real payoff potential

The Total Picture

The Total Picture